Claims

What is claimed is:

1. A computer-implemented method for performing memory management of an object in an object-oriented programming environment using smart pointers, comprising the steps of:

providing a base class common to all types of smart pointers;

providing at least one smart pointer comprising an object pointer for pointing to an object, the smart pointer having a next link for pointing to a subsequent smart pointer on a ring and a previous link for pointing to a previous smart pointer on the ring; and

providing a function for automatically converting a smart pointer to an object of a first class to a smart pointer to an object of a second class, wherein the first class and the second class share the common base class.

2. The method according to claim 1, comprising the steps of providing single member test for determining if a selected smart pointer is the only member of the ring and providing a deletion means for deleting the object if the selected smart pointer is determined to be the only member of the ring.

3. A computer-implemented method of memory management, comprising the steps of:

providing a smart pointer for association with a memory-resident element, the smart pointer including a next pointer;

providing an assignment means for assigning the next pointer to point to the smart pointer thereby creating a linked list comprising the smart pointer; and

providing a comparison means for comparing the value of the next pointer to the value of the memory location of the smart pointer in which the selected next pointer is included, whereby a determination can be made if the ring contains more than one smart pointer.

4. The method according to claim 3, wherein the method comprises the step of providing a common base to the smart pointer.

5. The method according to claim 3, wherein the element is an object in an object-oriented programming environment.

6. The method according to claim 5, wherein the smart pointer includes an object pointer for pointing to the object.

7. The method according to claim 3, wherein the linked list comprises a ring.

8. The method according to claim 3, wherein the smart pointer includes a previous pointer.

9. The method according to claim 8, comprising a step of providing an assignment means for assigning the previous pointer to point to the smart pointer, thereby creating a bi-directional, doubly-linked list.

10. The method according to claim 9, wherein the linked list comprises a ring.

11. The method according to claim 3, comprising a step of providing a deletion means for deleting the memory-resident element associated with the smart pointer if the value of the next pointer of the smart pointer is equal to the value of the memory location of the smart pointer in which the next pointer is included.

12. The method according to claim 3, wherein the smart pointer includes a first smart pointer, and wherein the method comprises the step of providing an attachment means for attaching a second smart pointer associated with the memory-resident element to the linked list.

13. A computer-implemented method of memory management, comprising the steps of:

providing a linked list comprising a smart pointer associated with a memory-resident element, the smart pointer including a next-pointer for pointing to the smart pointer; and

providing a comparison means for comparing the value of the memory location of the smart pointer to the value of the next-pointer of the smart pointer, to provide a determination whether the linked list contains only the smart pointer.

14. The method according to claim 13, comprising the step of providing a deletion means for deleting the memory-resident element when the value of the memory location of the smart pointer equals the value of the next-pointer of the smart pointer, whereby the memory assigned to the memory-resident element is released when no further reference can be made to the memory-resident element.

15. The method according to claim 13, wherein the element is an object in an object-oriented programming environment.

16. The method according to claim 15, wherein the smart pointer includes an object pointer for pointing to the object.

17. The method according to claim 13, wherein the linked list comprises a ring.

18. The method according to claim 13, wherein the smart pointer includes a previous pointer.

19. The method according to claim 18, comprising a step of providing an assignment means for assigning the previous pointer to point to the smart pointer, thereby creating a bi-directional, doubly-linked list.

20. The method according to claim 19, wherein the linked list comprises a ring.

21. The method according to claim 13, comprising a step of providing a deletion means for deleting the memory-resident element associated with the smart pointer if the value

of the next-pointer of the smart pointer is equal to the value of the memory location of the smart pointer in which the next-pointer is included.

22. The method according to claim 13, wherein the smart pointer includes a first smart pointer, and wherein the method comprises the step of providing an attachment means for attaching a second smart pointer associated with the memory-resident element to the linked list.

23. A computer-implemented method of memory management, comprising the steps of:

providing a linked list comprising a first smart pointer and a second smart pointer each associated with a memory-resident element, the first smart pointer including a first next-pointer for pointing to the second smart pointer and the second smart pointer including a second next-pointer for pointing to the first smart pointer; and

providing a comparison means for comparing the value of the memory location of a selected smart pointer giving up its association with the memory-resident element to the value of the next-pointer of the selected smart pointer, to provide a determination whether the linked list contains only the selected smart pointer.

24. The method according to claim 23, comprising the step of providing a deletion means for deleting the memory-resident element when the value of the memory location of the selected smart pointer equals the value of the next-pointer of the selected smart pointer, whereby the memory assigned to the memory-resident element is released when no further reference can be made to the memory-resident element.

25. The method according to claim 23, wherein the linked list comprises a ring.

26. The method according to claim 23, wherein the first smart pointer and the second smart pointer each include a previous pointer.

27. The method according to claim 26, comprising a step of providing an assignment means for assigning the previous pointer of the first smart pointer to point to the second smart pointer and for assigning the previous pointer of the second smart pointer to point to the first smart pointer, thereby creating a bi-directional, doubly-linked list.

28. The method according to claim 27, wherein the linked list comprises a ring.

29. The method according to claim 23, comprising a step of providing a deletion means for deleting the memory-resident element associated with the selected smart pointer if the value of the next-pointer of the selected smart pointer is equal to the value of the memory location of the selected smart pointer.

30. The method according to claim 23, comprising the step of providing an attachment means for attaching a third smart pointer associated with the memory-resident element to the linked list.

31. The method according to claim 23, comprising the step of providing a common base to the smart pointers.

32. The method according to claim 23, wherein the element is an object in an object-oriented programming environment.

33. The method according to claim 32, wherein the first smart pointer and the second smart pointer each include an object pointer for pointing to the object.

34. The method according to claim 32, wherein the first smart pointer is associated with a first object of a first class and the second smart pointer is associated with a second object of a second class, and wherein the method comprises the step of providing a conversion means for providing automatic conversion between the first smart pointer and the second smart pointer.